Appendix LSI設計者のための UML 2.0表記法ガイド

塚田雄一

ここではUMLの表記法について紹介する．UML 2.0では12種類の表記法が規格化されている．LSI設計では，例えばユースケース図やクラス図，シーケンス図などが用いられる．ハードウェア設計者にとってはなじみの状態遷移図やフローチャートに相当する表記法もある．（編集部）

UMLは“unified modeling language”，すなわち「統一モデリング言語」の略です．「言語」というとC/C++やJavaなどのプログラム・コードを思い浮かべるかもしれませんが，UMLはテキスト・ベースの記述言語ではなく，モデリング言語です．システム分析やシステム設計などを行うためのもので，システムをモデル化（抽象化）して図示するための表記法を規定しています．さらに，UMLによるシステム設計ではオブジェクト指向の考えかたを適用できます．そのため，UMLは「オブジェクト指向のための統一モデリング言語」と呼ばれています．

UMLの最新バージョンは2.0で，UML 2.0では12種類の表記法が存在します（**表1**）．ここで注意していただきたいのは，UMLはあくまでも図の書きかたにすぎないということです．まずは表記法を覚え，表記したいときに必要に応じて使いたい図を描きます．つまり，UML表記法をどのように使うかが重要となってきます．

● LSI開発に適用すると何がうれしいのか？

まず最初に，UMLをLSIの開発に適用した場合，何がうれしいのかについて説明します．

第1に，ドキュメントとしての効果があります．UMLはソフトウェア開発の標準言語です．よって，UMLをLSIの開発に導入すると，ソフトウェア技術者もレビューに参加できます．また，担当者が変わったときの引き継ぎなども容易になります．そして，英訳も比較的容易に行えるため，海外とのコミュニケーションにも役立ちます．

第2に，UMLはシステム全体の構造を把握するのに便利な表記法です．UMLを使うと，システムの構造を考えながら設計しやすくなります．その際，オブジェクト指向を利用することで，似たような処理を一つにまとめられます．こうすると，むだな処理の記述を減らせます．また，実装後の回路面積を小さくすることに役立ちます．

第3に，ほかのモジュールなどとの関連が記されているため，仕様変更などの際にほかのモジュールへの影響を把握しやすくなり，開発効率や設計品質が向上すると考えられます．

以下では，UML 2.0のモデルを紹介します．とくに，LSIの開発において重要と思われるモデルの記述を詳細に説明していきます．一方，使用頻度が少ないと思われるモデルの記述は，細かい説明を省略しています．

● ユースケース図：要求される機能をユーザの視点で表現

システムに要求される機能をユーザの視点から見た形で表現するのがユースケース図です．

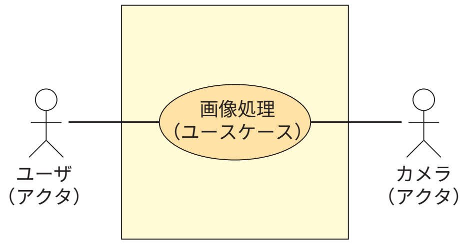
1）ユースケース図の基本表記

ユースケース図は，「アクタ（ユーザ）」と「ユースケース（システムの機能）」の二つから成り立っており，アクタである「ユーザ」とユースケースである「システム」の対話関係をモデル化します．

アクタは人の形で示します．そして，ユースケースはだ円で示します．また，システムの外部と内部を明確にするための境

表1　UML 2.0がサポートしている図

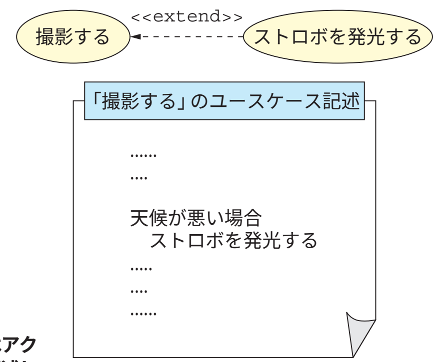
名　前	説　明
ユースケース図	システムのユーザ要求を表現
クラス図	部品（クラス）の内容と関係を表現
オブジェクト図	部品（オブジェクト）の関係を表現
シーケンス図	時間軸のオブジェクトの流れを表現
ステートマシン図	オブジェクトの状態や遷移を表現
アクティビティ図	処理の流れを表現
パッケージ図	グループ化した単位の関係を表現
コミュニケーション図	オブジェクト間のメッセージのやり取りを表現
タイミング・チャート	時間軸の変化を表現
コンポーネント図	部品の構造を表現
コンポジット・ストラクチャ図	クラスやコンポーネントの内部構造を表現
配置図	システムの物理的な配置を表現



(a) ユースケース図

図1 ユースケース図

ユースケース図はシステムのユーザ要求を表現する．(a)はアクタ(ユーザ，カメラ)とユースケース(画像処理)の関係を記述している．(b)のユースケース図の拡張記述は，すでに存在するユースケースを拡張する際の表現である．例えば，ユースケース記述のシナリオで「天候が悪い場合」という特定条件で，「撮影する」というユースケースから<<extend>>を使用して，「ストロボを発光する」と拡張した記述が可能となる．



(b) ユースケース図の拡張記述

ユースケース番号：
ユースケース名：
概　要：
アクタ　：
事前条件：
事後条件：
トリガ　：
基本シナリオ：
①処理1
②処理2
③処理3
④処理4
⑤処理5
代替シナリオ：
異常シナリオ：
コメント：

(c) ユースケース記述の例

界線として長方形を書きます．最後に，アクタとユースケースの関連を線で結びます．例えば，ユーザ，カメラというアクタから画像処理というユースケースを記述した例を**図1(a)**に示します．モデリングを行ううえでは，アクタからの視点で考えるということがポイントになります．

すでに存在するユースケースを拡張して別のユースケースを表現する場合，<<extend>>や<<include>>などを使用します．例えば<<extend>>は，特定の条件が成立したときのみ実行する記述です．これをユースケース図の拡張記述と呼びます．例えば，「天候が悪い」という特定条件の場合，「ストロボを発光する」という処理の拡張表現は**図1(b)**のようになります．特定条件は，次項で説明する「ユースケース記述」を使って示します．

2) ユースケース記述

ユースケース図は見てもわかるとおり，非常に単純なものです．したがって，ユーザとのレビューを行う際には補足説明が必要になります．その補足説明に当たるものがユースケース記述です．ユースケース記述にはとくに決まった記述方法があるわけではありません．ユースケース記述では，「アクタは何であるか？」，「実行するまでの前提条件は何か？」，「実行するためのトリガは何か？」といったことを記述します．また，実行のためのシナリオなども記述します．

ユースケース記述ではシナリオ内の分岐(代替)異常をもれなく記述することが重要になります．**図1(c)**に，記述フォーマットの例を示します．なお，ユースケース記述において，**図1(c)**の項目のすべてをかならず記述しなくてはならないというわけではありません．

● クラス図：部品の構成や関連を表現する最重要モデル

クラス図は，部品の構成や関連などを表現するのに用いられます．クラス図はUMLの中でもいちばん基本となるモデルです．そもそもクラスとは，部品や集合を示します．学校にたとえると，生徒(という部品)と先生(という部品)が集まって一つのクラスになっているのと同じです．

1) クラス図の基本表記

クラス図は，三つの領域で区切られた長方形で表現されます(**図2(a)**)．そしていちばん上の領域に「クラス名」を記述し，中央の領域に「属性(データ)」を記述します．そして，いちばん下の領域に「操作(処理)」を記述します．ちなみに操作では，関数であることを示すため，ことばの末尾に()を付けます．一つのシステムは複数のクラスから構成されます．個々のクラスはほかのクラスと何らかの関連を持っています．関連は線で表現します．

2) クラスの内部詳細記述

次にクラスの内部詳細記述について説明します(**図2(b)**)．クラスには可視性という項目があります．クラスを使う側から見える場合は属性や操作の前に「+」を記述し，クラスを使う側から見えない場合は「-」を記述します．属性にはデータ型と初期値を記述できます．そして操作にはパラメータ(引き数)と戻り値(リターン値)が記述可能です．

3) 誘導，関連クラス

クラス間の関連は，通常，実線で接続されます．ただし，あるクラスから別のクラスに誘導可能な場合は「→(矢印)」で記述します(**図3(a)**)．

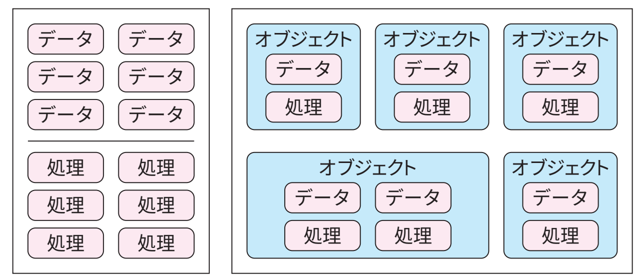
クラスとクラスの接続において詳細記述を行いたい場合は，**図3(b)**のような関連クラス記述を利用します．

COLUMN オブジェクト指向

そもそもオブジェクト指向とは何なのでしょうか？ 従来のソフトウェア開発では，機能に注目した開発手法が採用されていました．これに対してオブジェクト指向は，「物（オブジェクト）」に注目した開発手法です．機能を組み合わせてシステムが構築されるのではなく，「物」を組み合わせてシステムが構築されていると考えるのが，オブジェクト指向の考えかたです（図A-1）．

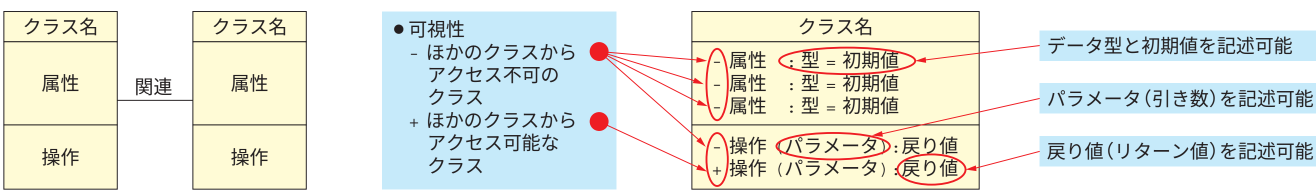
「データ」，「処理」などの部品を組み合わせれば，大規模なシステム構築も容易に行えます．そのため，オブジェクト指向は大規模システムの開発に向いています．また，部品で構成されているので，システムの変更や再利用の際に，必要な部品のみ変更すればよいことになり，品質確保や効率化にも優れています．

オブジェクト指向の世界では，変数（data）のことを「属性」と呼びます．また，関数（処理）のことを「操作」と呼びます．はじめは聞き慣れないことばかもしれませんが，属性と書かれていたときは変数のことであり，操作と書かれていたときは関数のことであると理解してください．UMLはオブジェクト指向を表記する記述が含まれているため，UMLを使ってオブジェクト指向設計を行うことが可能です．



（a）従来のソフトウェア開発　（b）オブジェクト指向

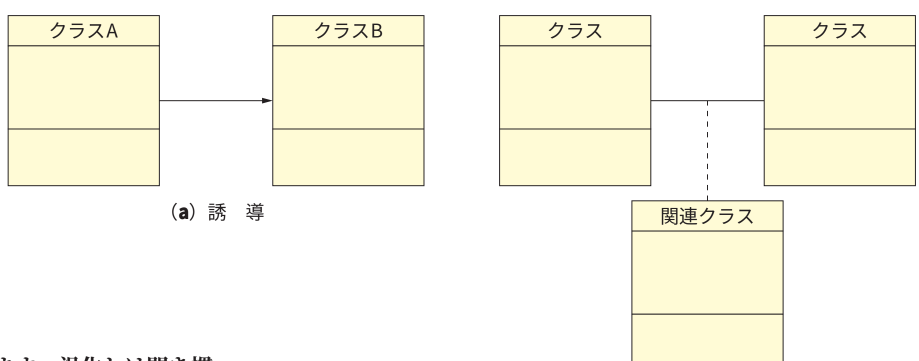
図A-1　オブジェクト指向の考えかた



（a）クラス図　（b）クラス内部詳細記述

図2　クラス図

クラス図は，クラス，オブジェクトなどの部品の構成やそれらの関連を表現する．(a)のようにクラスは，データを表現する「属性」と関数を表現する「操作」から構成される．(b)のクラス内部の詳細記述では，可視性を表現できる．ほかのクラスからアクセス可能なものには＋を記述し，アクセス不可能なものには－を記述する．ただし，オブジェクト指向では基本的に，属性には－を推奨し，ほかのクラスからのアクセスでは＋の操作を返して行うことを推奨している．また，属性には「型」と「初期値」を，操作には受け渡す「パラメータ」と「戻り値」を記述できる．



（a）誘　導　（b）関連クラス

図3
誘導と関連クラス
(a)の「誘導」は，インスタンスを行って使用する際の表現．クラスAがクラスBを誘導（インスタンスして使用）する際は，→（矢印）で表現する．(b)の「関連クラス」は関連の詳細を記述する表現．クラスとクラスの接続に詳細記述を行いたい場合（例えば，通信のプロトコル記述など），関連クラス記述を用いて表現する．

4）汎　化

クラス図には「汎化」という概念があります．汎化とは聞き慣れないことばだと思いますが，「汎用化」と言い換えると理解しやすいかもしれません．例えば，USBクラスとTCP/IPクラスの二つのクラスが存在していたとします．このとき，両方とも通信という汎用性のあるクラスにまとめる際に用います（図4(a)）．

5）集　約

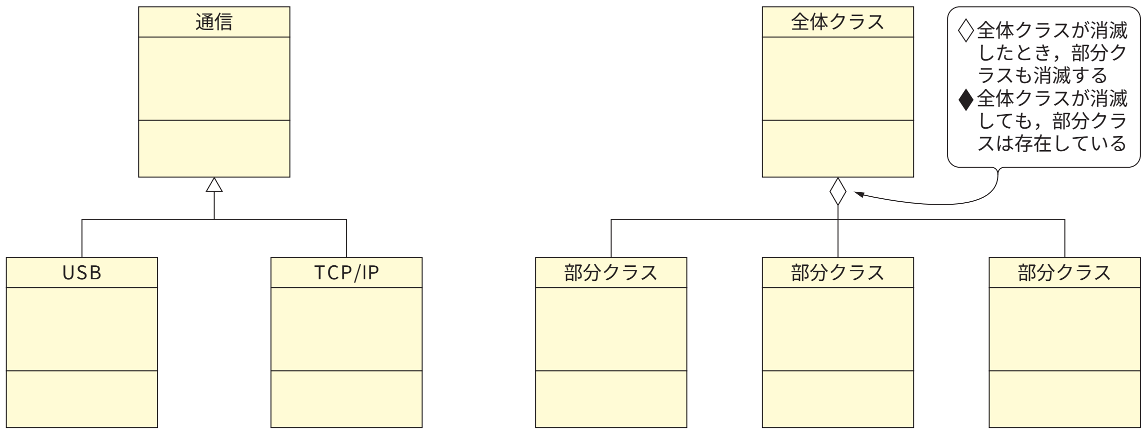
集約は，いくつかのクラスをまとめて一つのクラスにすることをいいます．これは，ある複数の部分クラスがあり，それらを集約して一つの全体クラスにする際に用います（図4(b)）．

6）依　存

あるクラスがあるクラスの要素を必要とする場合，つまりそのクラスに依存する場合，図5(a)のような表記でその関係を表現します．

7）インターフェース

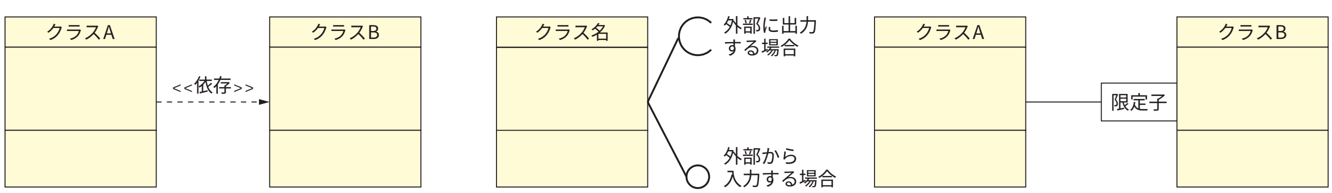
インターフェースとは，通信の際のやり取り（プロトコル）などの表現に使用します．表記は図5(b)に示すとおりです．



(a) 汎　化　　(b) 集　約

図4　汎化と集約

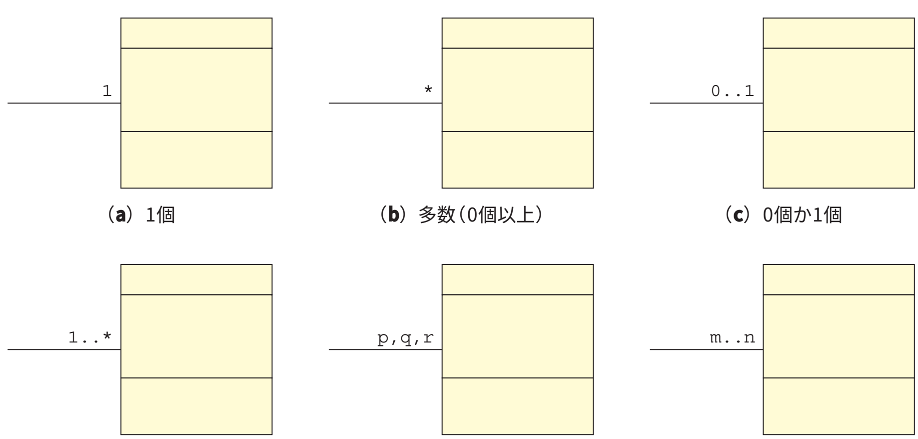
(a)の「汎化」を用いると，TCP/IPクラスとUSBクラスから汎用性のある通信クラスを作成できる．「特化」という表現もあり，通信クラスからTCP/IPクラスとUSBクラスを作成することができる．(b)の「集約」は，クラスを集めて表現する記述．部分クラスを集めて全体クラスを表現できる．このとき，◇で集約すると，全体クラスが消滅したときに部分クラスも消滅する．◆で集約すると，全体クラスが消滅したときにも部分クラスは存在したままになる．



(a) 依　存　　(b) インターフェース　　(c) 限定子

図5　依存，インターフェース，限定子

(a)の「依存」は，あるクラスの要素が必要とされる場合の表現である．クラスAがクラスBの要素を使用する場合，クラスAはクラスBの要素を知っている必要がある．しかし，使用される側のクラスBはクラスAの要素を知っている必要はない．その関係を「依存」と呼ぶ．(b)の「インターフェース」は，外部とのやり取りに使う表現である．「外部に出力する場合」と「外部から入力する場合」を記述できる．(c)の「限定子」はインスタンスを特定する際の表現である．クラスAからクラスBへアクセスする際に，かならず限定子のインスタンスを使用するといった記述が可能となる．



(a) 1個　　(b) 多数（0個以上）　　(c) 0個か1個

(d) 多数（1個以上）　　(e) 複数個（離散値）　　(f) 範囲指定

図6
クラス図のインスタンス記述

インスタンスが1個の場合は「1」と記述する．多数存在する場合は「*」と記述する．0個か1個などのOR表現では「0..1」と記述する．1個以上の場合は「1..*」と記述する．複数個（離散値）の場合は「p,q,r」と記述する．範囲を指定する場合は「m..n」と記述する．

8）限定子

限定子はインスタンスを特定する際に用います（図5(c)）．例えば，あるクラスと関連するときに，特定の関連を割り付ける際などに使用します．

9）インスタンスの種類

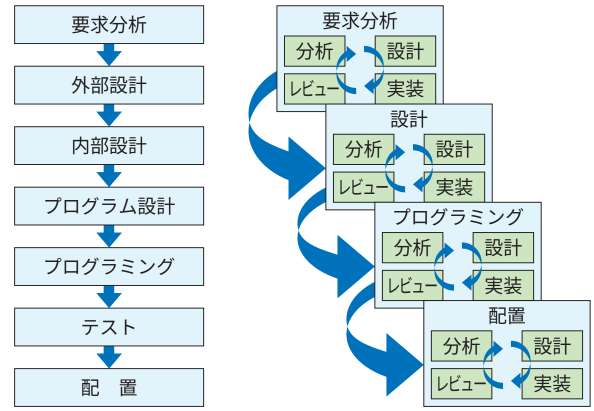
クラス図では，さまざまなインスタンスの種類を表現できます．例えば，「1個の（インスタンスの）記述」，「多数の記述」，「0個か1個の記述」，「複数個（離散値）の記述」，「範囲指定の記

COLUMN ウォータ・フォール・モデルとスパイラル・モデル

システム開発の手順として，「ウォータ・フォール・モデル」や「スパイラル・モデル」があります(図B-1)．

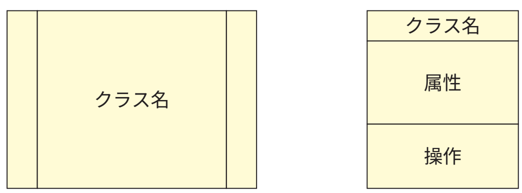
ウォータ・フォール・モデルは，要求分析→外部設計→内部設計→プログラム設計→プログラミング(コーディング)→テスト→配置といったように，工程が上から下へと滝のように進むモデルです．つねにレビューを行い，後戻りによるむだを省くことが重要になります．

これに対して，スパイラル・モデルはシステムの開発工程を，要求分析，設計，プログラミング，配置に分類し，各工程で分析→設計→実装→レビューを繰り返しながら開発を進める手法です．オブジェクト指向は，オブジェクトの単位でシステムを分割するため，スパイラル・モデルが適していると言われています．



図B-1 システム開発の手順
(a) ウォータ・フォール・モデル　(b) スパイラル・モデル

図7 アクティブ・クラスとパッシブ・クラス
状態を持ったクラスをアクティブ・クラスと呼ぶ．アクティブ・クラスは後述するステートマシン図を含んだクラス図になる．これに対して，状態を持たない通常のクラスをパッシブ・クラスと呼ぶ．



(a) アクティブ・クラス　(b) パッシブ・クラス

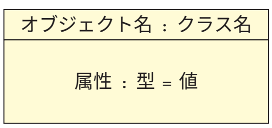


図8 オブジェクト・クラス
オブジェクト図は，オブジェクト(クラスのインスタンス)を表現する．「オブジェクト名:クラス名」と記述する．

述」などがあります(図6)．

10) パッシブ・クラスとアクティブ・クラス

クラスの中には状態を持ったものがあります．これを「アクティブ・クラス」と呼びます．つまり，アクティブ・クラスは後の項で説明する「ステートマシン図」を含んだクラス図となります．一方，通常のクラスを「パッシブ・クラス」と呼びます(図7)．

● オブジェクト図：クラスのインスタンス間の関係を表現

オブジェクト図は，オブジェクト(クラスのインスタンス)を表現するためのものです．クラス図にも記述されますが，シーケンス図にも記述されます．

オブジェクトの記述方法は，「オブジェクト名:クラス名」を下線付きで表現するだけです．また，オブジェクト図は下部に「属性:型＝値」という形式でオブジェクトの持つ属性の現在の値を明記することもできます(図8)．

● シーケンス図：オブジェクトの流れを時間軸で表現

シーケンス図は，オブジェクトやクラスの流れを時間軸に沿って表現するために用います．特定のシナリオを実現するために，オブジェクト(クラス)どうしの交信を記述することができます．

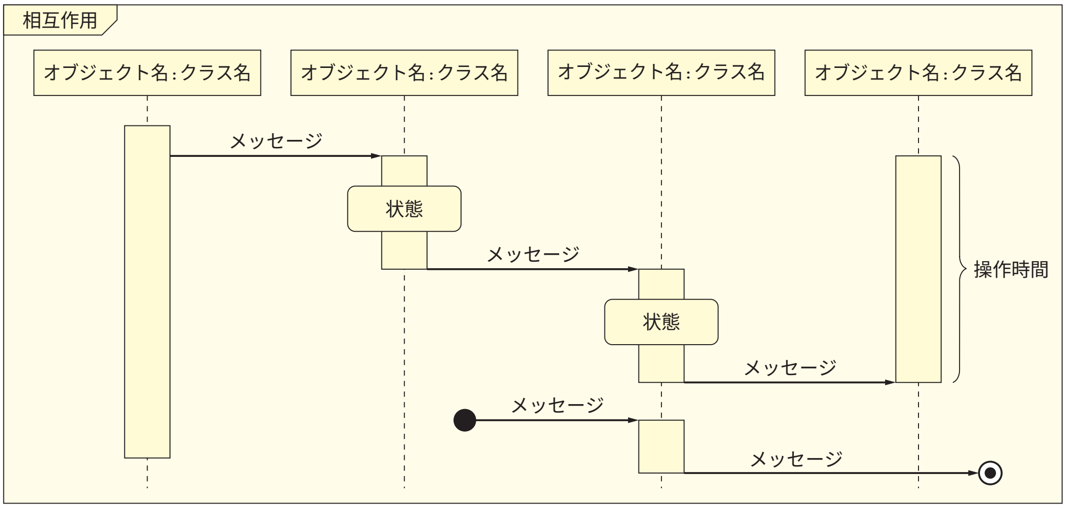
オブジェクト(クラス)どうしの交信を記述するためには，その前にユースケース図，シナリオ，オブジェクト図(クラス図)を作成しておく必要があります．シーケンス図はメッセージの流れを示すものですが，縦軸に時間を取り，横軸にオブジェクト(クラス)を並べます．そして，オブジェクト(クラス)の操作を実行している時間(活性区間)を長方形で記述します(図9(a))．

メッセージは4種類あります(図9(b))．すなわち，メッセージの応答を待ち，同期しながら交信する「同期メッセージ」，メッセージの応答を待たない「非同期メッセージ」．交信が開始されるときの「開始メッセージ」，交信が消滅するときの「消滅メッセージ」です．

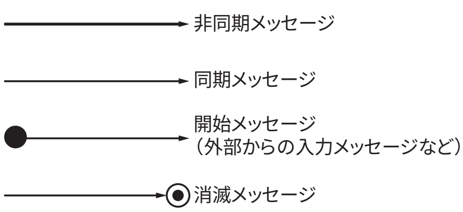
また，シーケンス図はUML 2.0になってから条件分岐を記述できるようになりました(図9(c))．例えばある条件のときにある処理を実行し，そのほかの条件(else)のときに別の処理を実行するなどのシーケンスを表現できます．

● ステートマシン図：状態遷移図に相当

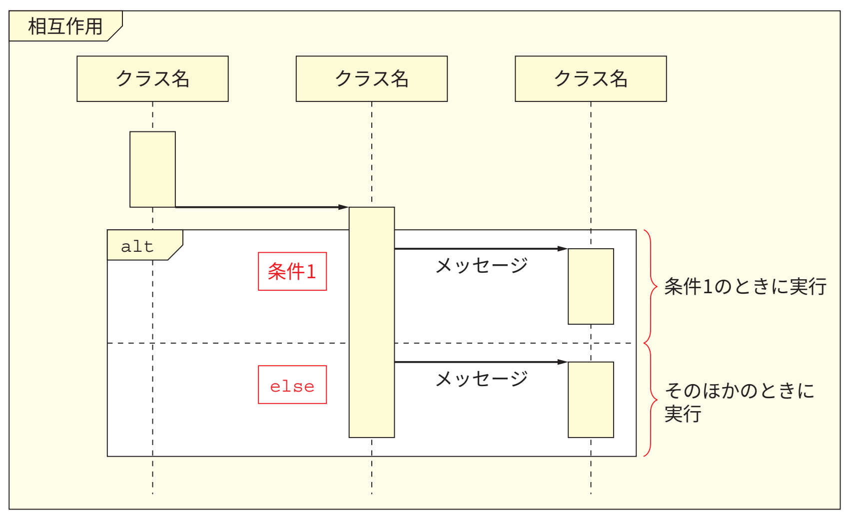
ステートマシン図は状態とその間の遷移を表現します．つま



(a) シーケンス図



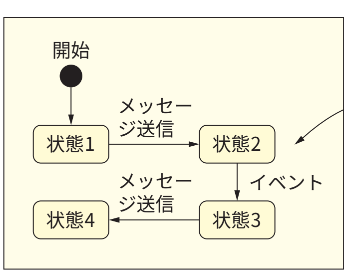
(b) メッセージ



(c) 条件分岐記述

図9　シーケンス図

(a)のシーケンス図は，時間軸のオブジェクトの流れを表現する．オブジェクトの状態，オブジェクト間のメッセージのやり取りを記述する．シーケンス図のメッセージには，(b)の4種類がある．シーケンス図では，(c)のように`alt`を使用して条件分岐を表現できる．条件1の場合は上記部分を実行し，`else`(そのほか)の場合は下記部分を実行するなどの記述が可能である．



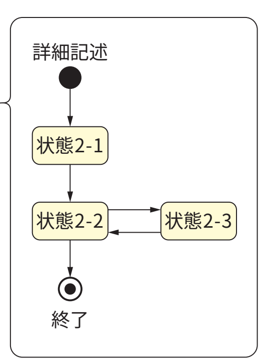


図10　ステートマシン図

ステートマシン図は，システムの状態やその遷移を表現する．状態が遷移する際のメッセージなどを記述できる．

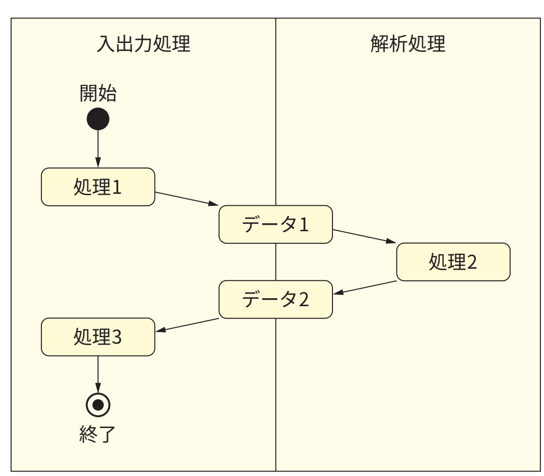


図11　アクティビティ図

アクティビティ図は処理の流れを表現する．図では「入出力処理」と「解析処理」の二つのレーンが存在し，各処理およびその間のデータの受け渡しを記述している．

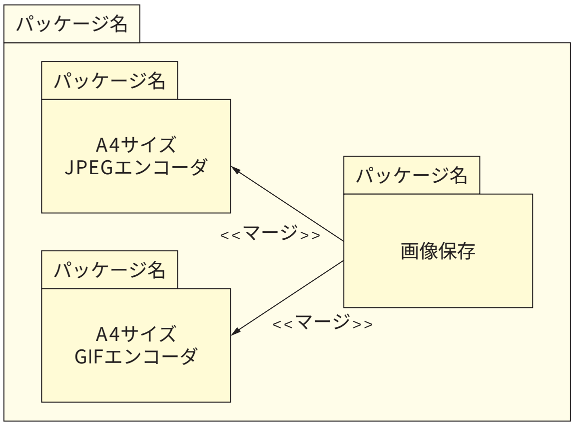


図12　パッケージ図

パッケージ図は，グループ化した単位の関係を表現する．各パッケージ間の関連などを記述できる．

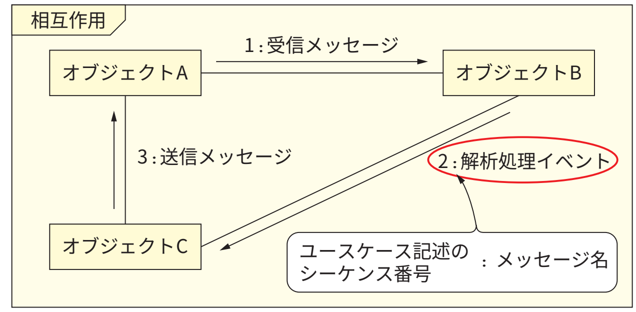


図13　コミュニケーション図

コミュニケーション図は，オブジェクト間のメッセージのやり取りを表現する．メッセージは，「ユースケース記述のシーケンス番号：メッセージ名」などと記述する．

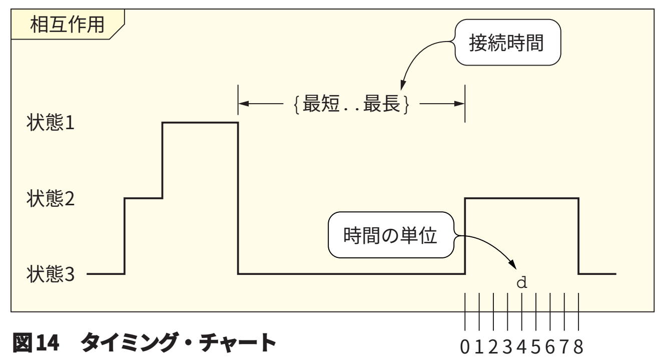


図14　タイミング・チャート

タイミング・チャートは，時間軸に対する状態の変化，あるいは時間制限などを表現する．持続時間や時間の単位などを記述できる．

り，システムがどのような状態にあるのか，そして状態はどのように変化していくのかを示します．

一つの状態は，角の取れた長方形で表します．一つの状態の中に，階層的に下位の状態や状態遷移を記述することもできます．また，開始ポイントには●を，終了ポイントには◉を書きます(**図10**)．前項で紹介したクラス図のアクティブ・クラスは状態を持つことが可能であり，アクティブ・クラスはステートマシン図と関連付けされているという表現になります．

● アクティビティ図：フローチャートに相当

アクティビティ図は，処理の流れを表現するために用いられます．これはフローチャートに似ています．機能を分割する表現として「レーン」を記述します．例えば，入出力処理と解析処理といった単位で処理を分割します．

処理やデータは角のとれた長方形で記述します．また，処理の流れの開始ポイントには●を，メッセージの最後には◉を書きます(**図11**)．

● パッケージ図：システム中の部品をグループ化して表現

パッケージ図は，システムの中にどのような部品が存在しているかを表現するために用いられます．パッケージはモデルの要素の入れ物のようなもので，どのような部品をパッケージ化するか(グループとしてまとめるか)を記述するためのものです．

パッケージ図では，大きな長方形の左肩に小さな長方形を乗せ，その中に「パッケージ名」を記述します．パッケージ図の中にはクラス図を記述することができます．また，階層的にパッケージ図の中にさらに下位のパッケージ図を記述することも可能です(**図12**)．

● コミュニケーション図：メッセージのやり取りを表現

コミュニケーション図は，オブジェクト間のメッセージのやり取りを表現するために用いられます．長方形でオブジェクトを示し，その間を直線で結んでオブジェクト間のメッセージを記述します．メッセージのフォーマットは，「ユースケース記述のシーケンス番号：メッセージ名」になります(**図13**)．

● タイミング・チャート：時間の概念を表現

タイミング・チャートは時間の概念を表現するために用いられます．波形に似た表現であり，その中に時間的な制約を記述できます．例えば，ある状態の持続時間(最短，最長)などを記述可能です(**図14**)．時間制約が重要なLSI開発の仕様書には必須の表現であると言えます．

● コンポーネント図：クラスやオブジェクトの構造を表現

コンポーネント図は，部品の構造を表現するために用いられます．例えば，クラス構造やオブジェクト構造などをどのように梱包するかを示します．また，各コンポーネントの間はイン

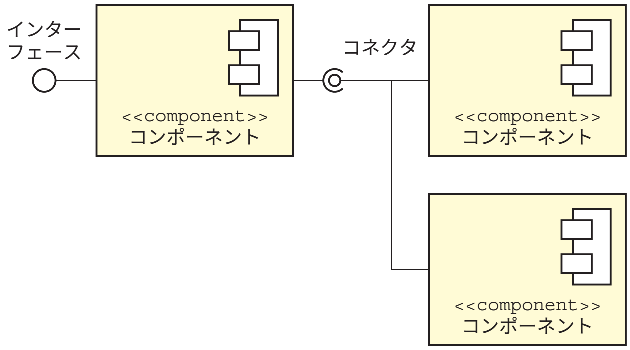


図15　コンポーネント図

コンポーネント図は，部品の構造を表現する．各コンポーネントの内容，コンポーネント間の関連を記述できる．

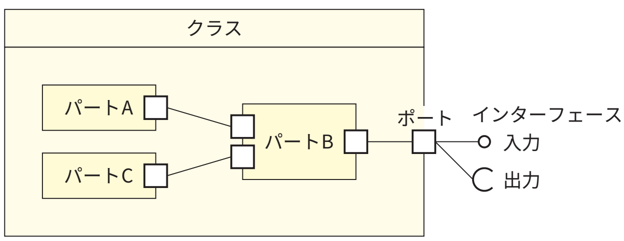


図16　コンポジット・ストラクチャ図

コンポジット・ストラクチャ図は，クラスやコンポーネントの内部構造を表現する．各パートが外部と通信するための「ポート」と，外部クラスと通信するための「インターフェース」などを記述できる．

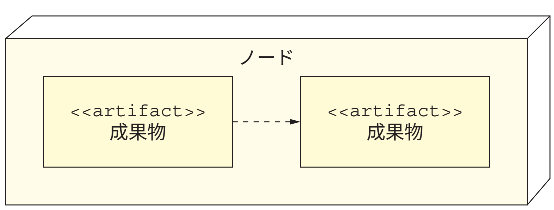


図17　配置図

配置図は，システムの物理的な位置関係を表現する．ノードの内部に，成果物として<<artifact>>を記述する．各<<artifact>>間の関連も表現できる．

ターフェースやコネクタなどによって接続されます(**図15**)．

● コンポジット・ストラクチャ図：部品の内部構造を表現

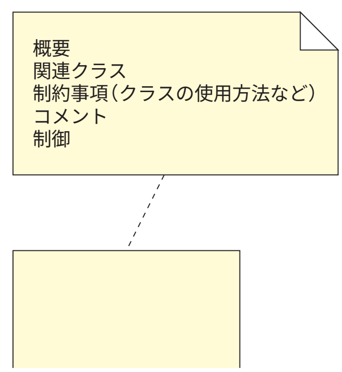
コンポジット・ストラクチャ図は，クラスやオブジェクトの内部構造を表現するために用いられます．SystemCで構造を表現する場合と同じく，各クラスが外部とやり取りする際にポートを介して接続される点が特徴です．クラス図のところでも紹介しましたが，入力，出力のインターフェース記述も行えます(**図16**)．

COLUMN　ノート記述とステレオタイプ

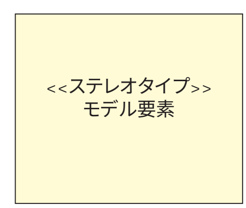
UMLの任意の図の中にコメントを書き込む場合，ノートを利用します．ノートの記述は，角の折れた長方形で示されます(**図C-1(a)**)．使用方法は，コメントの対象となる図記号からノートに引き出し線を引くだけです．

また，UMLを拡張する際にはステレオタイプという機能を用います．これは<<ステレオタイプ>>と記します(**図C-1(b)**)．

ユースケース図のところで紹介した<<include>>なども，実はステレオタイプの表現の一つです．これは，クラスを拡張した「include」というモデルが存在しており，そのモデルをステレオタイプを使用して記述したものです．本誌2004年12月号，pp.88-96の記事「LSIをUMLとSystemCでモデリングする」で紹介したSoC向け拡張プロファイルは，システムLSIを記述するためにSoCModule，SoCPortなどのモデルを用意しておき，ステレオタイプを使ってそのモデルを使用して記述しています．



(**a**) ノートの記述



(**b**) ステレオタイプ

図C-1　ノートとステレオタイプ

ノートは，制御やコメントを記述できる．ステレオタイプは<<ステレオタイプ>>で示され，クラスを拡張した表現になる．

● 配置図：物理的な位置関係を表現

配置図は，システムの物理的な位置関係を表現するために用いられます．つまり，実装フェースにいちばん近い表現となります．成果物は<<artifact>>で記述し，各成果物の関連を表現します(**図17**)．

つかだ・ゆういち
キャッツ(株)